Grid Cell Formation

Alison C. Ebaugh, August 2008

August 18, 2008

Abstract

In this report we describe the formation of grid cells through the collection of self organized head direction cell input. We will discuss a simple model that indicates that the directional information encoded by head direction cells can be manipulated to produce a grid cell that has similar preferences such as the head direction cell input. At the conclusion of this report we will show the relationship between velocity, inhibitory firing rate and the formation of even grid field spacing.

1 Introduction

The study of space in the brain has been developed extensively since the initial discovery of place fields over three decades ago. It has now been established by multiple different experiments that spatial information of a given environment is processed through different neural mechanisms in the hippocampus and the entorhinal cortex. This system acts as a cognitive map created by the internal cues that represent the velocity and direction of the animal. These different neural mechanisms include a system of three different cells, head direction cells, grid cells, and place cells. This report focuses on the connection between head direction and grid cells. It is our hypothesis that grid cells form through a process of self organization by integrating input from random head direction cells.

2 Background

The hippocampus and entorhinal cortex operate together to produce these neural mechanisms that govern the rat's perception of space. Figure 1 **b** depicts the subfields of the hippocampus, such as dentate gyrus (DG), subiculum (S), and the hippocampal fields CA1 and CA3.

Head direction cells are found in multiple areas of the brain, such as the anterior dorsal nucleus of the anterior thalamus and the dorsal sector of the caudal lateral dorsal thalamic nucleus.These cells encode directional information of animal's environment and respond to a preferred direction of the head. The cells maximally fire at a specific angle that corresponds to the particular cell's preferred head direction. As the animal turns its head away from the preferred head direction the firing rate of that cell decreases slowly. Experimental data that supports this type of characteristic can be seen in figure 1 **(b)** and figure 2. In figure 1 **(b)** the black curves represent the trajectories of the rat as it explores an environment and the red dots represent the position of the rat when the head direction cell spiked.

The other key component of spatial representation is grid cells. Grid cells are mainly found in the medial entorhinal cortex. As an animal explores an environment, grid cells fire at select locations. The firing fields of each grid cell produce an equilateral triangular grid over the surface of the environment explored by the

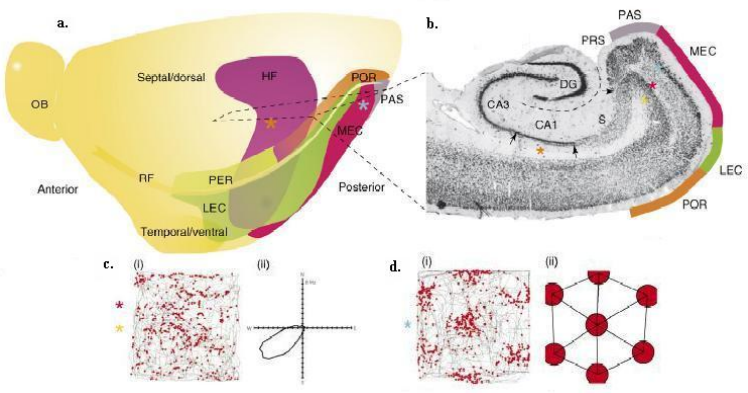

Figure 1: This figure the includes a picture of the left hemisphere of rat where HF and the MEC denotes the **hippocampal formation** and the **medial entorhinal cortex** respectively, **(a)** and **(b)**. Electrode readings from layer II and V of the MEC depict experimental results of head direction cells, **(b)** and a grid cell **(d)**. This figure was obtained from **(1)**.

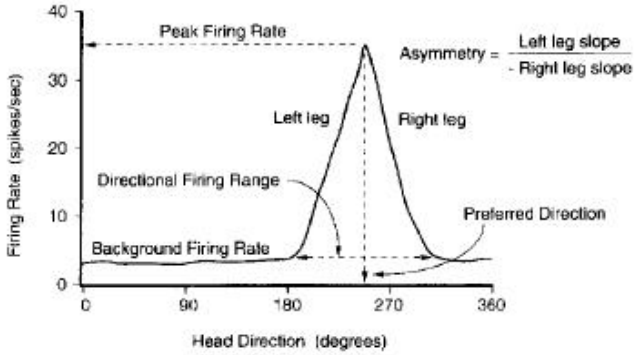

Figure 2: Head direction cell different characteristics.This figure was obtained by **(2)**.

animal. This grid is used as a universal coordinate system as the animal internally locates itself in the environment. These observations can be seen in the experimental data in figure 1 **(c)**.

3 Simple Model of Grid Cell Formation

To test our hypothesis concerning the formation of grid cells we created a simple model that simulates the neural interactions between a set of given head direction cells and one grid cell. In our simulation we created a virtual rat that randomly explores a circular environment. As the rat explores the bounded area the program simulates the neural responses to the different head directions the rat is experiencing, denoted as θ. These different head directions of the rat determine if any of the individual head direction cells will fire. The collection of these firing rates provides the excitatory input to the lone grid cell. For a visual representation of this model refer to figure 3.

After the rat has completed a specified amount of time exploring the circular environment the program calculates the summation of all firing rates of each head direction cell over time. These firing rates are

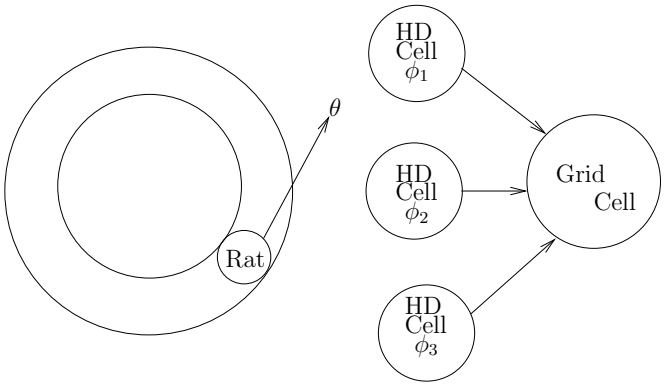

Figure 3: Simplified process of grid cell formation.

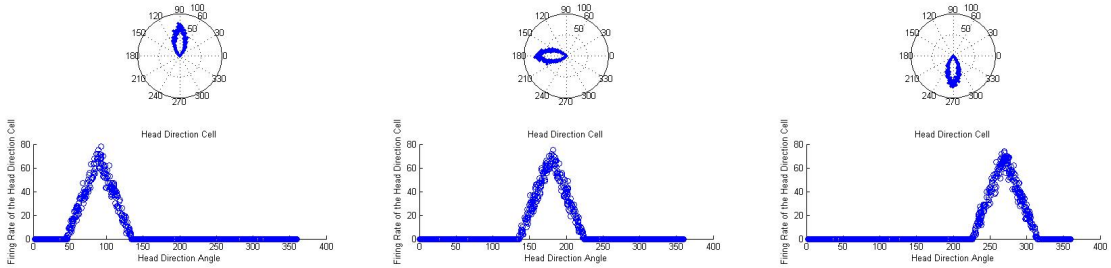

Figure 4: Head direction cells with preferences of $\phi_1 = 90$, $\phi_2 = 180$, and $\phi_3 = 270$ degrees.

plotted against all the head directions of the rat. We found that this yielded a grid cell.

For example consider three different head direction cells with assigned preference of, $\phi_1 = 90$, $\phi_2 = 180$, and $\phi_3 = 270$ degrees. The firing rate of each head direction cell compared to all recorded head directions of the rat is displayed in the results of the simulation shown by figure 4. The grid that is formed from these three head direction cells can be found in figure 5. The code for this simulation can be found in Appendix A.

4 Integrate and Fire Model

We extended this basic model of grid cell formation to an integrate and fire network of grid cells. An integrate and fire network refers to a network of neurons where each cell's voltage is governed and continuously updated by the conductance of the particular cell. In this model the head direction cell provides the excitatory input that stimulates the grid cell network of synaptic connections that are nonplastic. The dynamics of every cell in the network is governed by the three following differential equations at every timestep,

$$\tau_M V'(t) = V_{rest} - V(t) + g_E(V_E - V(t)) + g_I(V_I - V(t)) \tag{1}$$

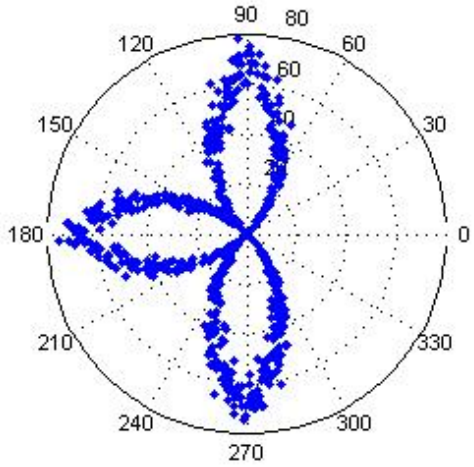

Figure 5: Grid cell that received input from three head direction cells.

$$\tau_E g'_E(t) = -g_E(t) + \sum_{i=1}^{N} \sum_{j}^{\infty} w_i^{hd} \delta(tr_{hd} - j) + \sum_{i=1}^{n} \sum_{j}^{\infty} w_i^E \delta(tr_E - j) \tag{2}$$

$$\tau_I g'_I(t) = -g_I(t) + \sum_{i=1}^{n} \sum_{j}^{\infty} w_i^I \delta(tr_I - j). \tag{3}$$

Where V represents the voltage of the cell, gE denotes the excitatory conductance and gI represents inhibitory conductance. The other parameters can be found in table 1. These differential equations are approximated utilizing the Backward Euler method.The code for this simulation can be found in Appendix B.

Table 1: Parameters for Integrate and Fire Differential Equations

Parameter	**Definition**
τ_M	decay time constant for membrane voltage
τ_E	decay time constant for excitatory conductance
τ_I	decay time constant for inhibitory conductance
w^{hd}	synaptic weights connecting the head direction cells to the grid cell network
w^E	synaptic weights of interconnections between excitatory grid cells
w^I	synaptic weights of interconnections between inhibitory grid cells
r_I	the firing rate of inhibitory input in Hz
r_{hd}	the firing rate of head direction cell input in Hz
N	the number of head direction cells
n	the number of grid cells in the network
V_{rest}	resting potential
V_E	excitatory reversal potential
V_I	inhibitory reversal potential

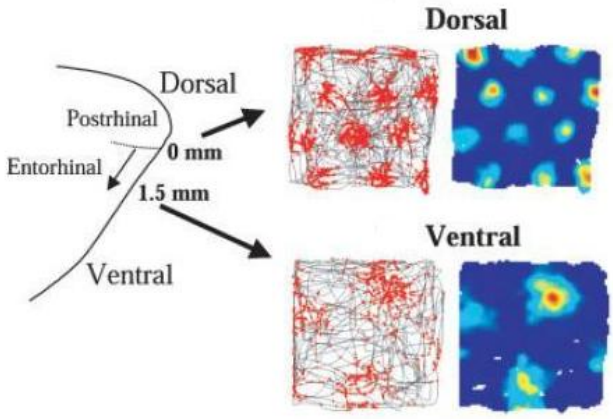

Figure 6: Different grid field spacing found in different locations of the entorhinal cortex. This figure was obtained from **(3)**.

5 Grid Field Spacing

The spacing between each grid field is dependent on the location of the cell in the entorhinal cortex. Neurons along the dorsal axis of the medial entorhinal cortex produce high subthreshold oscillations. Through experimental observations high subthreshold oscillations are correlated with finer grid cell spacing. Conversely, neurons along the ventral axis produce lower subthreshold oscillations and have been correlated with wider grid cell spacing. Some experimental results can be seen in the figure 6.

6 Velocity and Inhibitory Input

Currently we are working on introducing a varying inhibition rate that is dependent on the rat's velocity. As seen in equation (3) the inhibitory conductance of the network depends on the rate of inhibition, r_I in the network. It is our hope that the value of r_I can be tuned to regulate the voltage such that grid cell firing will only occur for a predefined spacing.

To test this observation we have created a program that the calculates the specific inhibition rate of the network at the given following parameters; velocity, grid field spacing and constant excitatory rate denoted as r_{hd}. This program analyzes the neural response of one excitatory grid cell that is receiving constant head direction input at a rate of r_{hd}. The program utilizes the bisection method to approximate the value of r_I when the voltage of a grid cell reaches threshold value and ultimately spikes at the appropriate grid field spacing. This result is then checked against a program that analytically derives the voltage at the specific inhibition rate, r_I, velocity and rate of excitatory rate, r_{hd}.

Thus far our findings are consistent with our prediction. In figure 7 the plot shows the relationship between the provided inhibition rate, denoted r_I and velocity. The general trend of the graph depicts an inverse relationship between velocity and the rate of inhibition for a grid spacing of 1/3cm.

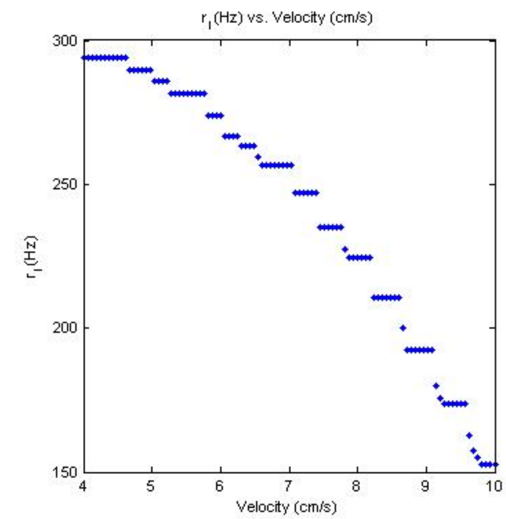

Figure 7: The relationship between inhibition rate and velocity of the rat for a grid field spacing of 1/3 cm.

7 Conclusion

In conclusion we are much closer to fully understanding the formation of grid cells and ultimately the neural mechanisms that impact our interpretation of position in space. Through our observation that velocity and inhibition rate of the network are related, it is our hope that grid field spacing will become evenly spaced on the surface level of rat's environment and match experimental data. This will prove that grid cell formation depends on the collection of head direction cell input and the velocity of rat.

8 References and Acknowledgements

(1) Moser and Witter.(2006) Spatial Representation and the Architecture of the Entorhinal Cortex. *TRENDS in Neuroscience*,29,671-678
(2) Taube.(1998)Head Direction Cells and the Neurophysiological Basis for a Sense of Direction. *Progress in Neurobiology*,55,225- 256
(3) Hasselmo, Giocomo, and Zilli.(2007) Grid Cell Firing May Arise From Interference of Theta Frequency Membrane Potential Oscillations in Single Neurons. *HIPPOCAMPUS*,17,1252-1271

Special acknowledgements to the following individuals of Department of Computational and Applied Mathematics of Rice University: Steve Cox, Anthony Kellems,Eric Libby and Kathryn Ward

This work was partially supported by NSF REU Grant DMS-0755294 and the following individuals from the

9 Appendix A

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Name: HeadDirection0
%
% Usage: HeadDirection0(paw,hdnum,timeatHD)
%
% Example: HeadDirection0(.1,12,100)
%
% Where paw: is how fast the rat is plotted to the screen
%       hdnum: is the number of head direction cells in the system
%       timeatHD: is the time spent at each head direction
%
% This function plots a rat running through a maze and the calculates the
% angle inwhich the rat is moving. This function also calculates the rate
% inwhich the specific head direction cells are stimulated by a given head
% direction of the rat. The foundation of this code was created from the 'reproducing'
% figure 3 in Samsonovich & Ascoli, 2005, Learning and Memory, 12:193-208
% and Eric Libby.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function HeadDirection0(paw,hdnum,timeatHD)
close all

% draw the maze
hfig = figure;
hold on;
t = [0:.1:100];
maze_y = sin(t)+ 100;
maze_x = cos(t)+ 100;
plot(maze_x,maze_y,'black')
axis off



% Initialize the position of the rat
radius= 0.05;
disp(' ')
disp(' Please use the mouse to start the rat')
xy_0= ginput(1); %The user chooses the initial position of the rat

% Creates the dot that represents the rat in the maze
h=rectangle('position',[xy_0-radius,[2 2]*radius],'curvature',1,'facecolor','r','edgecolor','w');
loc=get(h,'position'); %--current location
xy=loc(1:2);

%Initialize Parameters
itMax = 500;  %Max iterations
curHD = zeros(itMax,1); %initializes the rat's head direction
%theta = [1:360];
alp = 1-0.125;
sig = 0.1;
v = [0 0];
set(h,'position',loc);% initial position
```

```
% Sets HD cell preference
%listofHD = [30:30:360];
%listofHD = sort(rand(1,hdnum)*360)
listofHD = [90 180 270];

%Simulates the rat running around in the maze
for it=1:itMax

    try     % try assigns random values to the next_xy until next_xy is inside the maze

        for mm=1:40,
            ksi = randn2(sig);
            next_xy = xy + alp*v + ksi;
            next_xy
            if inmaze1(next_xy,maze_x,maze_y,radius),
                error('ok')
            end;
            if mm==20,
                v = [0 0];
            end;
        end;
        fprintf('couldn''t find a ksi\n');
        return;

    catch   % accept the move

        % Checks that next_xy is still in the maze and updates kinematic
        % state of rat accordingly
        b = bump(xy, next_xy, maze_x, maze_y,radius);
        if any(b),
            next_xy= b; % reassigns the next_xy so that the rat stays within the maze
            v = [0 0];
        else
            v = alp*v + ksi; % updates velocity of the rat
        end;

        % Moves the rat to the next_xy position
        figure(1)
        moveto(next_xy,h,radius,'blue'); % draws the line segment from xy to next_xy
        pause(paw)

        %Calculates and stores the angle,theta, of the xy position
        theta =  atan2(next_xy(2)-xy(2),next_xy(1)-xy(1));
        if theta < 0,
            theta = 2*pi+theta;
        end

        curHD(it) = theta;
        theta=  theta*(180/pi);
```

```
        % Calculates the rate of spikes at the current head direction,
        % theta at a given timestep
        [SpikeRate, E] = GridSim1(timeatHD,it,theta,listofHD);

        %Stores the rate of spikes for each current HD for plotting
        HDavg(it,:) = SpikeRate;

        G(:,it)= E;
        xy = next_xy; % Sets the next position as the initial

    end  % try
end  % for itany(b)

%Plots the each HD cells and the corresponding grid cell

%Sums the time blocks for each individual head direction
for i=1:length(curHD)

    Eavg(i)=sum(G((i-1)*timeatHD+1:i*timeatHD));

end

plottingHD(hdnum,HDavg,curHD,Eavg);

return;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: inmaze1
% Usage: result = inmaze(next_xy,maze_x,maze_y,radius)
%
% Where result: indicates if the rat's next position is
%               inside or outside the maze
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function result = inmaze1(next_xy,maze_x,maze_y,radius);
r1= .8*radius;
xy=next_xy+r1;
if (100-xy(1))^2 + (100-xy(2))^2 < 1
    result = 1; %inside the maze
else
    result=0; %outside the maze
end;
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: bump
% Usage: new_xy = bump(xy,next_xy,maze_x,maze_y,radius)
%
% Where: new_xy: is the new value of x and y that keep
```

```
%                the rat inside the maze
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function new_xy = bump(xy,next_xy,maze_x,maze_y,radius);

%Checks if initial values of x and y are in the maze
if not(inmaze1(xy,maze_x,maze_y,radius)),
    disp('outside');
    new_xy=xy; % resets the next_x = x, rat doesn't move
    return;
end;

%Displacement of the orginal position and next
d=next_xy-xy;

%Calculates the number of moves in the displacement
nmov=ceil(sqrt(d*d'))./radius;
d=d/nmov;

%Checks if each move is in the maze
for k=1:nmov,
    new_xy=xy+k*d;
    if not(inmaze1(new_xy,maze_x,maze_y,radius)),
        new_xy=new_xy-d./5;
        while not(inmaze1(new_xy,maze_x,maze_y,radius)),
            new_xy=new_xy-d./5;
        end;
        disp('second return');
        return;
    end;
end;
new_xy =[0 0]; % if there are no bumps
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: moveto
% Usage: moveto(xy, h, rad, col);
%
% This function plots the line segment
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function moveto(xy, h, rad, col);
loc=get(h,'position');
line([loc(1)+rad, xy(1)+rad],[loc(2)+rad, xy(2)+rad],'color',col);
loc(1:2)=xy;
set(h,'position',loc);
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: randn2
% Usage: val = randn2(sig);
```

```
% Calculates a random value and multiples it by sig
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function val = randn2(sig);
val = randn(1,2)*sig;
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% GridSim1
% usage:   GridSim1(timeatHD,it,theta,listofHD)
%
% exmaple: GridSim1(10,1,35,7)
%
% timeatHD: is the time step at every head direction input
% hdnum: is the total number of head direction cells
% it: is the given iternation of the rat
% theta: the current head direction expressed in degrees
%
% Returns boolean matrix (columns are different cells and rows are time steps at a given head direction
% a 1 means it fires, a 0 means it didn't
% HDstatelist : Head direction neuron
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [SpikeRate,E] = GridSim1(timeatHD,it,theta,listofHD)

% Calculates if the HD cells fired at every timestep at the given theta
for j = 1:timeatHD

    HDfirerate = HDconvert(listofHD,theta);

    HDstate = (HDfirerate>rand(size(HDfirerate)));

    HDstatelist(j,:) = HDstate';

end

%This is the rate of each HD cell firing at that individual head direction
SpikeRate = sum(HDstatelist);
E = sum(HDstatelist,2);
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: HDconvert
%
% Usage: HDfirerate = HDconvert(listofHD,curHD)
%
% This function calculates the probability that a
% current head direction will cause a head direction
% cell to fire
%
% Returns a fire rate for each HD cell
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function HDfirerate = HDconvert(listofHD,curHD)

maxrate = .7; % maximum prob. firing a time step
width = 45; % width of firing distribution
Hdfirerate = zeros(size(listofHD));

% Naive method for mapping loop of angles onto a line
if (curHD+width)>360
    uwrap = 1;
    amax = width-(360-curHD);
else
    uwrap = 0;
    amax = curHD+width;
end

if (curHD-width)<0
    dwrap = 1;
    amin = 360-(width-curHD);
else
    dwrap = 0;
    amin = curHD-width;
end

for i=1:length(listofHD)

    temp = listofHD(i);

    if uwrap == 1
        if ((temp<amin) & (temp>amax))
            HDfirerate(i) = 0;
        elseif (temp<= amax)
            HDfirerate(i) = ((width-(360-curHD+temp))/width)*maxrate;
        elseif (temp>= amin)
            HDfirerate(i) = ((width-(abs(curHD-temp)))/width)*maxrate;
        end
    elseif dwrap == 1
        if ((temp>amax) & (temp<amin))
            HDfirerate(i) = 0;
        elseif (temp<= amax)
            HDfirerate(i) = ((width-abs(curHD-temp))/width)*maxrate;
        elseif (temp>= amin)
            HDfirerate(i) = ((width-(360-temp+curHD))/width)*maxrate;
        end
    else
        if ((temp>= amin) & (temp<= amax))
            HDfirerate(i) = ((width-(abs(curHD-temp)))/width)*maxrate;
        else
            HDfirerate(i) = 0;
        end
    end
end
```

```
return;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function plottingHD(hdnum,HDavg,curHD,Eavg)

for j=1:hdnum,
    figure(j+1)
    subplot(2,1,1)
    polar(curHD,HDavg(:,j),'.')
    x = (180/pi)*curHD;
    y = HDavg(:,j);
    subplot(2,1,2)
    scatter(x,y)
    xlabel('Head Direction Angle')
    ylabel('Firing Rate of the Head Direction Cell')
    title('Head Direction Cell');
end

% To combine Head Direction Cells
% This takes the sum of the rows of matrix H, basically suming the activity
% that occurs at every current head direction per time step
Eavg'
figure(hdnum+2)
polar(curHD,Eavg','.');
title('Grid Cell')
return;
```

10 Appendix B

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: HD_grid2
%
% Usage: HD_grid2(T,dt,hdnum,timeatHD)
%
% Example: HD_grid2(10,1,7,10)
%
% Where T: is the total time of the simulation in sec
%       dt: is the timestep in (ms)
%       hdnum: is the number of head direction cells in the system
%       timeatHD: is the time spent at each head direction
%
% This function plots a rat running through a maze and the calculates the
% angle inwhich the rat is moving. This function also calculates the rate
% inwhich the specific head direction cells are stimulated by a given head
% direction of the rat. The foundation of this code was created from the `reproducing'
% figure 3 in Samsonovich & Ascoli, 2005, Learning and Memory, 12:193-208
% and Eric Libby.
%
% Intergrate and Fire network is used by Kathryn Ward:
% no plasticity modeled
% input: excitatory HD cells provided by Alison
```

```
% input given to excitatory cells within a network of MEC E and I cells
% goal: see MEC cells fire in grid patterns
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function HD_grid2(T,dt,hdnum,timeatHD)
%stdp4 Input
% T: total time of simulation (sec)
% dt: timestep (ms)

close all

Tms = T*1000;   %T is ms

Ne_inj = hdnum;  %number of MEC excitatory cells
Ne_net = 2;  %number of excitatory neurons in CA3 network
Ni_net = 1;  %number of inhibitory neurons in CA3 network

%parameters for the MEC network
tauM_E = 20;    %exc. membrane voltage decay constant (ms)
tauM_I = 20;
Vrest_E = -70;  %exc. resting potential (ms)
Vrest_I = -70;
% Eex = 0;  %excitatory reversal potential (mV)
Ein = -70;
Vth_E = -54;    %exc. threshold potential (mV)
Vth_I = -54;
Vreset_E = -60; %exc. reset potential (mV)
Vreset_I = -60;

tau_gE = 5;   %excitatory conductance decay constant (ms)
tau_gI = 5;

%Set weights from Head direction cells to Grid E cells, column=pre, row=post
wMax = .1;
Winj_to_net = wMax*sprand(Ne_net,Ne_inj,1);
figure(2)
spy(Winj_to_net)
title('Winj to net')

%Set weights within Grid cell network
WEtoE = .1*sprand(Ne_net,Ne_net,.5);  %excitatory onto excitatory weights
WEtoE = WEtoE - spdiags(diag(WEtoE),0,Ne_net,Ne_net);   %no cell synapses onto itself
WItoE = .1*sprand(Ne_net,Ni_net,0.5);
WEtoI = .1*sprand(Ni_net,Ne_net,.5);

WItoI = spalloc(Ni_net,Ni_net,0);
W = [WEtoE,WItoE; WEtoI,WItoI];
figure(3)
spy(W)
title('W net')
```

```
%Initiate vars for the MEC network
v_E = Vrest_E*ones(Ne_net,1);  %voltage for MEC E cells
v_I = Vrest_I*ones(Ni_net,1);
v = [v_E;v_I];
gI_E = 0*v_E;  %inhibitory conductance of MEC E cells
% gI_I = 0; %no I-I connections
gE_E = gI_E;   %excitatory conductance of MEC E cells
gE_I = 0*v_I;   %excitatory conductance of MEC I cells
gE = [gE_E;gE_I];

%constants to be used in while loop
gIbot = tau_gI + dt;
gEbot = tau_gE + dt;
quot_I = tauM_I/dt;
quot_E = tauM_E/dt;

%Assume all MEC cells are initially at rest
spikedEx_net = [];
spikedIn_net = [];

%Setup for HD portion of code
MECtoMon = 1:Ne_net+Ni_net; %MEC cells to monitor
NumMon = length(MECtoMon);
subdim = sqrt(NumMon+1);
jnum = ceil(Tms/dt)
MECspikes = spalloc(Ne_net+Ni_net, jnum, 5*(Ne_net+Ni_net)*jnum);

listofHD = [10 30 60 45 90 180 270]
%sort(rand(1,hdnum)*360)   %head direction preferences for HD cells
NumSpikedNet = 0;

%Plot maze for rat's movements
radius_maze = 100; %in cm
spacing = linspace(0,2*pi,1000);
maze_y = sin(spacing)+ radius_maze;
maze_x = cos(spacing)+ radius_maze;

%This figure monitorers the movement of the rat and records the location of
%the individual spikes
figure(1)
for k=1:NumMon+1
    subplot(subdim,subdim,k)
    plot(maze_x,maze_y)
    axis equal
    axis off
    if k>1
        title(['MEC cell ' num2str(k-1)])
    else
        title('rat')
    end
    hold on
```

```
end


% Initialize the position of the rat
subplot(subdim,subdim,1)
radius= 5*0.01;
disp(' ')
disp(' Please use the mouse to start the rat')
xy_0= ginput(1); %The user chooses the initial position of the rat

paw = .1;

% Creates the dot that represents the rat in the maze
h=rectangle('position',[xy_0-radius,[2 2]*radius],'curvature',1,'facecolor','r','edgecolor','w');
loc=get(h,'position'); %--current location
xy=loc(1:2);

%Initialize Parameters for the rat
alp = 1-0.125;
sig = 0.1;
vel = [0 0];
set(h,'position',loc);% initial position
number = Tms/dt;
disp('entering loop')
tic

j=1;
while j*dt < Tms
    %rat moves...HD cells fire
    try     % try assigns random values to the next_xy until next_xy is inside the maze

        for mm=1:40,
            ksi = randn2(sig);
            next_xy = xy + alp*vel + ksi;
            if inmaze1(next_xy,maze_x,maze_y,radius),
                error('ok')
            end;
            if mm==20,
                vel = [0 0];
            end;
        end;
        fprintf('couldn''t find a ksi\n');
        return;

    catch   % accept the move

        % Checks that next_xy is still in the maze and updates kinematic
        % state of rat accordingly
        b = bump(xy, next_xy, maze_x, maze_y,radius);
        if any(b),
            next_xy= b; % reassigns the next_xy so that the rat stays within the maze
```

```
    vel = [0 0];
else
    vel = alp*vel + ksi; % updates velocity of the rat
end;

% Moves the rat to the next_xy position
subplot(subdim,subdim,1)
moveto(next_xy,h,radius,'blue'); % draws the line segment from xy to next_xy
pause(paw)

%Calculates and stores the angle,theta, of the xy position
theta =  atan2(next_xy(2)-xy(2),next_xy(1)-xy(1));
if theta < 0,
    theta = 2*pi+theta;
end

curHD(1,j) = theta;

theta=  theta*(180/pi);

timesteps_at_HD = ceil(timeatHD/dt);   %in effect, forcing timeatHD to be a multiple of dt

%Make sure time does not pass Tms for current theta
if (j+timesteps_at_HD)*dt >=Tms
    timesteps_at_HD = floor(Tms/dt) - j;
end

d = next_xy - xy;
step = d / timesteps_at_HD;
for k=1:timesteps_at_HD
    HDfirerate = HDconvert(listofHD,theta);
    HDstate = (HDfirerate>rand(size(HDfirerate)));
    xy = xy + step;

    spiked_inj = find(HDstate);   %list of HD cells spiking at current timestep

    %conductances decay
    gItop_E = tau_gI*gI_E;
    gI_E = gItop_E/gIbot;

    gEtop = tau_gE*gE;
    gE = gEtop/gEbot;
    gE_E = gE(1:Ne_net);
    gE_I = gE(Ne_net+1 : Ne_net + Ni_net);

    %Add injected input to conductances
    gE_E = gE_E + sum(Winj_to_net(:,spiked_inj),2);

    %Add network input to conductances
    gI_E = gI_E + sum(WItoE(:,spikedIn_net - Ne_net),2);
    gE_E = gE_E + sum(WEtoE(:,spikedEx_net),2);
```

```
            gE_I = gE_I + sum(WEtoI(:,spikedEx_net),2);
            gE = [gE_E;gE_I];

            %Update voltages
            vtop_E = quot_E*v_E + Vrest_E + gI_E*Ein; % +gEx*Eex
            vbot_E = quot_E + 1 + gE_E + gI_E;
            v_E = vtop_E./vbot_E;

            vtop_I = quot_I*v_I + Vrest_I;
            vbot_I = quot_I + 1 + gE_I; % + gI_I
            v_I = vtop_I./vbot_I;

            v = [v_E;v_I];

            %find network spiked cells
            spikedEx_net = find(v_E >= Vth_E);
            spikedIn_net = find(v_I >= Vth_I) + Ne_net;
            spikedCells_net = [spikedEx_net; spikedIn_net];

            MECspikes(spikedEx_net,j) = theta;


            if ~isempty(spikedCells_net)
                v(spikedEx_net) = Vreset_E;
                v(spikedIn_net) = Vreset_I;
                v_E = v(1:Ne_net);
                v_I = v(Ne_net+1 : Ne_net+Ni_net);
                NumSpikedNet = NumSpikedNet + length(spikedCells_net);
            end

            figure(1)
            spikedMon = intersect(spikedCells_net,MECtoMon);
            for ct = 1:length(spikedMon)
                subplot(subdim,subdim,spikedMon(ct)+1)
                scatter(xy(1),xy(2),5,'r')
                hold on
            end
            %set next step
            j = j+1;

        end % this end is for the time_at_HD for loop

    end % this end is for the try

end % this end is for the while loop

% this plots the histrograms for each MEC grid cell
for i=1:Ne_net+Ni_net
    figure(i+1)
    fired_ind = find(MECspikes(i,:));
    [N,X]=hist(MECspikes(i,fired_ind),360);
```

```
    hist(MECspikes(i,fired_ind),360)
    title(['Number of Spikes per theta for MEC cell' num2str(i)])
    xlabel('Head Directions of the Rat')
    ylabel('Number of spikes')
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: inmaze1
% Usage: result = inmaze(next_xy,maze_x,maze_y,radius)
%
% Where result: indicates if the rat's next position is
%               inside or outside the maze
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function result = inmaze1(next_xy,maze_x,maze_y,radius);
r1= .8*radius;
xy=next_xy+r1;
if (100-xy(1))^2 + (100-xy(2))^2 < 1
    result = 1; %inside the maze
else
    result=0; %outside the maze
end;
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: bump
% Usage: new_xy = bump(xy,next_xy,maze_x,maze_y,radius)
%
% Where: new_xy: is the new value of x and y that keep
%                the rat inside the maze
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function new_xy = bump(xy,next_xy,maze_x,maze_y,radius);

%Checks if initial values of x and y are in the maze
if not(inmaze1(xy,maze_x,maze_y,radius)),
    disp('outside');
    new_xy=xy; % resets the next_x = x, rat doesn't move
    return;
end;

%Displacement of the orginal position and next
d=next_xy-xy;

%Calculates the number of moves in the displacement
nmov=ceil(sqrt(d*d'))./radius;
d=d/nmov;

%Checks if each move is in the maze
for k=1:nmov,
```

```
    new_xy=xy+k*d;
    if not(inmaze1(new_xy,maze_x,maze_y,radius)),
        new_xy=new_xy-d./5;
        while not(inmaze1(new_xy,maze_x,maze_y,radius)),
            new_xy=new_xy-d./5;
        end;
        disp('second return');
        return;
    end;
end;
new_xy =[0 0]; % if there are no bumps
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: moveto
% Usage: moveto(xy, h, rad, col);
%
% This function plots the line segment
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function moveto(xy, h, rad, col);
loc=get(h,'position');
line([loc(1)+rad, xy(1)+rad],[loc(2)+rad, xy(2)+rad],'color',col);
loc(1:2)=xy;
set(h,'position',loc);
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: randn2
% Usage: val = randn2(sig);
% Calculates a random value and multiples it by sig
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function val = randn2(sig);
val = randn(1,2)*sig;
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: HDconvert
%
% Usage: HDfirerate = HDconvert(listofHD,curHD)
%
% This function calculates the probability that a
% current head direction will cause a head direction
% cell to fire
%
% Returns a fire rate for each HD cell
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function HDfirerate = HDconvert(listofHD,curHD)

maxrate = .7; % maximum prob. firing a time step
width = 45; % width of firing distribution
```

```
Hdfirerate = zeros(size(listofHD));

% Naive method for mapping loop of angles onto a line
if (curHD+width)>360
    uwrap = 1;
    amax = width-(360-curHD);
else
    uwrap = 0;
    amax = curHD+width;
end

if (curHD-width)<0
    dwrap = 1;
    amin = 360-(width-curHD);
else
    dwrap = 0;
    amin = curHD-width;
end

for i=1:length(listofHD)

    temp = listofHD(i);

    if uwrap == 1
        if ((temp<amin) & (temp>amax))
            HDfirerate(i) = 0;
        elseif (temp<= amax)
            HDfirerate(i) = ((width-(360-curHD+temp))/width)*maxrate;
        elseif (temp>= amin)
            HDfirerate(i) = ((width-(abs(curHD-temp)))/width)*maxrate;
        end
    elseif dwrap == 1
        if ((temp>amax) & (temp<amin))
            HDfirerate(i) = 0;
        elseif (temp<= amax)
            HDfirerate(i) = ((width-abs(curHD-temp))/width)*maxrate;
        elseif (temp>= amin)
            HDfirerate(i) = ((width-(360-temp+curHD))/width)*maxrate;
        end
    else
        if ((temp>= amin) & (temp<= amax))
            HDfirerate(i) = ((width-(abs(curHD-temp)))/width)*maxrate;
        else
            HDfirerate(i) = 0;
        end
    end
end
return;
```

11 Appendix C

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: find_Irate
%
% Usage: = find_Irate(spacing,Irate_a, Irate_b, dt, vel)
%
% Example: find_Irate(1/3,200,300,1, 15)
%
% Where spacing:is the spacing in cm inbetween grid firing feilds
%       Irate_a: lower range of Irate
%       Irate_b: higher range of the Irate
%       dt: the timestep
%       vel: the velocity of the rat, fixed in cm/s
%
% This function calculates the inhibition rate of the network as the rat as it moves
% along a straight track.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Irate = find_Irate(spacing,Irate_a, Irate_b, dt, vel)
close all

Vth_E = -54;    %exc. threshold potential (mV)


%Initialize Parameters
Tms = (spacing/vel(1))*1000  %T is ms
tol = .001;
itmax = 1000;
HDrate = 35;% the max firing rate of the head direction cell

% finds the voltages at the Irate_a and Irate_b
v_a = find_V(Irate_a, Tms, dt,vel);
v_b = find_V(Irate_b, Tms, dt,vel);

% calculates how close the voltage is to the Vth, if f = 0 then a spiked
% has occurred
f_a =  Vth_E - v_a;
f_b = Vth_E - v_b;
i=1;

% this the start of the bisection method the finds the Irate when the cell
% spikes

% finds the correct range for the Irate_a and Irate_b for the intial
% interval for the bisection method
while f_a >0 & i<=itmax
    Irate_a = Irate_a - 5;
    v_a = find_V(Irate_a, Tms, dt,vel);
    f_a =  Vth_E - v_a;
```

```
    if i > itmax
        disp('itmax reached, no Irate_a' )
        return;
    end

    if Irate_a < 0
        disp('Irate_a is negative')
        return
    end
    i = i + 1;
end

j =1;
while f_b <0 & j <= itmax
    Irate_b = Irate_b + 5;
    v_b = find_V(Irate_b, Tms, dt,vel);
    f_b =  Vth_E - v_b;

    if j > itmax
        disp('itmax reached, no Irate_b' )
        return;
    end

    if Irate_b > HD_rate*1.5
        disp('Irate_b id 1.5 times bigger than HDrate')
        return
    end
    j = j + 1;
end

% initalizations for the bisection method
mid = (Irate_a + Irate_b)/2;
v_mid =  find_V(mid,Tms, dt,vel);
f_mid = Vth_E -v_mid;
k =1;

% bisection method
while (Irate_b-Irate_a)>tol

    if sign(f_a)*sign(f_mid)<0
        Irate_b =  mid;
    else
        Irate_a = mid;
        f_a = f_mid;
    end
    mid = (Irate_a + Irate_b)/2;
    v_mid =  find_V(mid,Tms, dt,vel);
    f_mid = Vth_E -v_mid;
    k = k +1;
end
Irate = mid;
```

```
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: find_V
%
% Usage: v_E = find_V(Irate_0, Tms, dt,vel)
%
% Example: v_E = find_V(300, 10, 1,10)
%
% Where Irate_0: the inhibitory rate of the network
%       Tms: time in ms
%       dt: the timestep
%       vel: the velocity of the rat, fixed in cm/s
%
% This function calculates the voltage at the time Tms and the give
% inhibition rate. Code provided by Katie Ward.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function v_E = find_V(Irate_0, Tms, dt,vel)
%Parameters
W_HD = 1.5;
W_I = .1;
HDrate = 35; % max firing rate of Head Direction cells (Hz)

close all

Ne_net = 1;  %number of excitatory neurons in Mec network
Ni_net = 0;  %number of inhibitory neurons in Mec network

%parameters
tauM_E = 20;    %exc. membrane voltage decay constant (ms)
Vrest_E = -70;  %exc. resting potential (ms)
% Eex = 0;  %excitatory reversal potential (mV)
Ein = -70;
Vth_E = -54;    %exc. threshold potential (mV)
Vreset_E = -60; %exc. reset potential (mV)

tau_gE = 5;   %excitatory conductance decay constant (ms)
tau_gI = 5;

%Parameters for a refractory period
tref_E = 5; %ref. period in ms


%constants to be used in while loop
gIbot = tau_gI + dt;
gEbot = tau_gE + dt;
quot_E = tauM_E/dt;

HD_tspike=ceil(1000/(HDrate*dt)); % timestep per spike
theta = 0;
```

```
%Initialize Parameters
Disp = [0 0];
xy = [0 0];
v_E = Vreset_E*ones(Ne_net,1);

I_tspike = ceil(1000/(Irate_0*dt)); %timestep per I spike
gI_E = 0*v_E;  %inhibitory conductance of MEC E cells
gE_E = gI_E;   %excitatory conductance of MEC E cells

%Assume all Mec cells are initially at rest
spikedEx_net = [];
spikedIn_net = [];

NumSpikedNet = 0;

%begin time in the steady state

%calculates the steady state of the conductances of g_I and g_E
dtI = 1000/Irate_0;
alphaI = exp(-dtI/tau_gI);
gIbar = W_I/(1-alphaI);

dtE = 1000/HDrate;
alphaE = exp(-dtE/tau_gE);
gEbar = W_HD/(1-alphaE);

gE_E = gEbar - W_HD;
gI_E = gIbar - W_I;
refracCells = tref_E;
spikedFlag = 0;

j=1;
while j*dt < Tms

    Disp(1)= vel*(dt/1000); % convert velocity to from cm/s to cm/ms
    xy = xy + Disp;

    %conductances decay
    gItop_E = tau_gI*gI_E;
    gI_E = gItop_E/gIbot;

    gEtop_E = tau_gE*gE_E;
    gE_E = gEtop_E/gEbot;


    %Add injected input to E conductances at the specific time
    if mod(j-1,HD_tspike)==0
        gE_E = gE_E + W_HD;
    end

    % Add velocity input to I conductance at the specific time
```

```
    if mod(j-1,I_tspike)==0
        gI_E = gI_E + W_I;
    end

    %Check if cell is refr.
    if refracCells > j*dt
        vtop_E = quot_E*v_E + Vrest_E; % +gEx*Eex
        vbot_E = quot_E + 1;
    else
        vtop_E = quot_E*v_E + Vrest_E + gI_E*Ein; % +gE_E*Eex
        vbot_E = quot_E + 1 + gE_E  + gI_E;
    end
    v_E = vtop_E./vbot_E;

    if spikedFlag > 0
        v_E = 0;
    end
    %
    %find network spiked cells
    spikedEx_net = find(v_E>Vth_E);

    if ~isempty(spikedEx_net)
        spikedFlag = 1;
        %Set refr. period
        refracCells = j*dt + tref_E;

    end
    j = j + 1;

end % this end is for the while loop

return
```